MAXIMAL ARC-DISJOINT FLOW IN A MULTICOMMODITY COMMON TERMINAL NETWORK

By

ASOK PULOOPPALLIL JACOB

INDUSTRIAL AND MANAGEMENT ENGINEERING PROGRAM

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

JULY, 1980

MAXIMAL ARC-DISJOINT FLOW IN A MULTICOMMODITY COMMON TERMINAL NETWORK

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

By

ASOK PULOOPPALLIL JACOB

to the

INDUSTRIAL AND MANAGEMENT ENGINEERING PROGRAM

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

JULY, 1980

IMEP - 1980 - M - JAC - MAX

CERTIFICATE

This is to certify that the present work on 'Maximal Arc-disjoint Flow in a Multicommodity Common Terminal Network' by Asok P. Jacob has been carried out under my supervision and has not been submitted elsewhere for the award of a degree.

(J.L. Batra)
Professor and Head
Industrial and Management Engg.Program
Indian Institute of Technology
Kanpur 208016.

July, 1980

ACKNOWLEDGEMENTS

I express my most heartfelt homage to my father who did not live to realize his ambition - that of witnessing his sons graduate. I dedicate this work to him in his loving memory as a token of my strive to fulfil his dreams.

Words fail to express my indebtedness to my affectionate mother and grandparents whose love and encouragement had always been a source of inspiration for me.

I express my deep sense of gratitude to my thesis supervisor Dr. J.L. Batra who not only has provided to me his invaluable suggestions, and unforgettable attachment, but also imbibed in me an extensive spectrum of academic rigour during the course of this work.

I am also thankful to Dr. S. Sadagopan and Mr. A.K. Ahuja for their valuable suggestions and constructive criticisms rendered during the course of this work.

It is indeed my pleasure to thank all my friends at I.I.T., Kanpur, whose amusing company made my two years of life in the campus a pleasant and memorable one.

I would like to thank Mr. C.M. Abraham for his neat and accurate typing and Mr. Buddhiram Kandiyal for his immaculate cyclostyling work.

Asok P. Jacob

CONTENTS

SYNOPSIS

This work considers the problem of determining the maximal arc disjoint flow in a multicommodity network with one source node for each commodity and a common terminal node for all the commodities. This problem is a special case of the general multicommodity flow problem and considers the restriction that more than one commodity should not flow on any arc. The problem is essentially that of partitioning the arcs of the network into as many sets as the number of commodities and allowing only one commodity to flow over each set, such that the sum of flows of all commodities from their source nodes to the common terminal is maximum.

The mathematical formulation of the problem is done as a mixed integer linear program. Two algorithms based on the well known branch and bound strategy are proposed for the solution of the problem. The two algorithms differ from each other only in the method of determining the upper bound of each node in the branch and bound tree.

Computer codes are written for the proposed algorithms in FORTRAN-10 and implemented on DEC-1090 time sharing computer system. Problems of size as large as 200 nodes, 300 arcs and 5 commodities were solved. Even for the largest size problems

solved, the average CPU time was less than 6 seconds in case of both the algorithms. The computational performances of the proposed algorithms for a large number of randomly generated networks of varied sizes are presented.

CHAPTER 1

INTRODUCTION

An important area in the study of networks is that of the maximum possible flow between two specified nodes of the network. In the broad sense, a flow on a graph is a way of sending objects from one vertex to another by travelling along the arcs in their directions. The problem of finding the maximum flow can be posed in almost every instance in which commodities flow from a source node to a destination node. Thus in traffic network, we may be interested in the maximum rate of traffic flow between two cities, in an electric power distribution network, we may be interested in the maximum power transmitted from a generating station or stations to a particular location; in a natural gas distribution network we may be interested in the maximum rate at which gas can be supplied to a particular consumer etc.

This is the most basic problem in network flow whose solution is readily available through a labelling procedure due to Ford and Fulkerson [3] or through the method of preflows by Karzanov [7]. There are two important extensions to the one source - one terminal flow problem.

The first is the multiterminal flow problem in which several nodes are designated as source-terminal (s,t) pairs,

with the same commodity flows through the network. For example, this is the case of a telephone network in which any unordered pair of n cities covered by the network may indeed serve as the (s-t) pair.

The second is the 'multicommodity' flow problems in which several commodities flow simultaneously from designated sources to designated terminals. The sources and the terminals may be different for different commodities. The problem is to find such a flow that the sum of the different flows from the respective sources to the respective terminals is maximum. For example, in multiproduct production - distribution systems, the production units and the customers serve as the source-terminal pairs and the distribution channels, the arcs of the network.

The problem considered in this work is a variation of the multicommodity flow problem where all the commodities flow to a common terminal node. Here an additional constraint that the flows are to be arc-disjoint is also imposed. This means no arc allows the flow of more than one commodity over it or each commodity will flow over mutually exclusive set of arcs. For example, in pipe line networks, where more than one commodity flow and these commodities originate from different sources and terminate in a common processing unit, there could be arc disjoint constraints due to technological reasons.

The same problem can also be posed as the problem of partioning 'the set of arcs into as many mutually exclusive sets as the number of commodities that are to pass through the network. Each of these subsets is assigned to one commodity. On such a subset, the maximal flow of the commodity for which it is assigned could be found using any single commodity, single source, single terminal maximal flow algorithm. Thus, it is required to find that partition which gives the maximum sum of flows.

We could also think of two other similar maximal flow problems in a network. One is the maximal arc disjoint flow problem in a multicommodity common source multiterminal network. The other is the maximal node disjoint flow problem in a multicommodity common terminal (or common source) network in which more than one commodity is not allowed to pass over any node(other than the common terminal (common source) node) in the network. Because of the underlying similarities, these problems could also be solved using the solution procedures suggested for maximal arc-disjoint flow problem in a multicommodity, common terminal network.

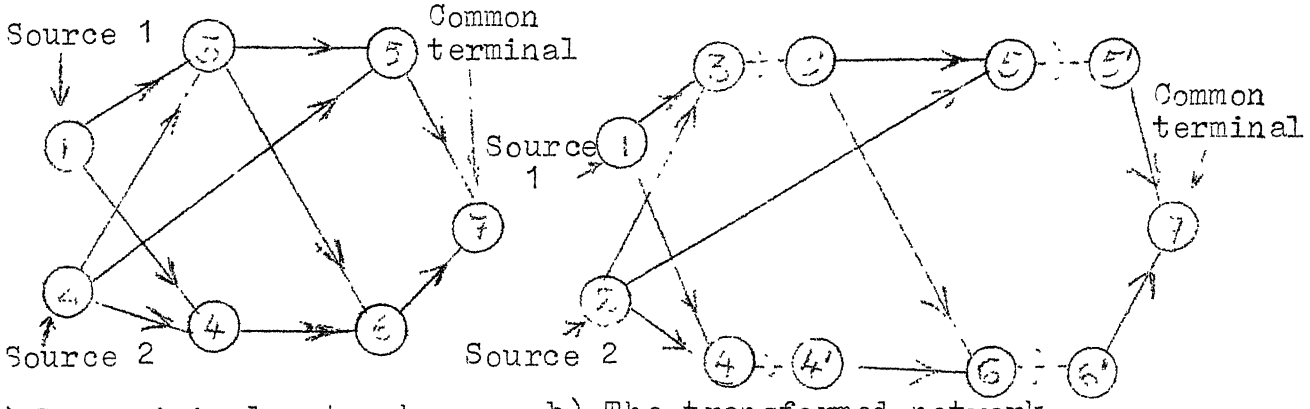
The common source problem could be solved using a branch and bound method similar to the one suggested for common terminal problem. Instead of connecting all the source nodes to a supersource, in the case of common terminal flow problem, all

the different sink nodes are connected to fictitious super-sink using arcs of infinite capacity.

The problem of node-disjoint flow is a further restriction of the arc disjoint flow problem. That is, all node disjoint flows are arc disjoint too. (But the converse of this statement that all arc disjoint flows are node disjoint is not true). Moreover all node disjoint flow problems may be converted to arc disjoint flow problems by a simple transformation given below.

For each node K (other than the source nodes and the common terminal node), add one new node K'. K and K' are connected by a directed arc of large capacity. All the incoming arcs to the corresponding node in the original network are incident upon the node K and all the outgoing arcs from the corresponding node in the original network will emerge out from the node K'. The source nodes and the common terminal node are kept undisturbed. Thus, if the original network had n nodes, m arcs and p commodities, the transformed network will have 2n-p-1 nodes and m+n-p-1 arcs. This transformation is illustrated in Fig. 1.

On the transformed network, the maximal arc disjoint flow will be the same as the node disjoint flow in the original network. This becomes obvious if we consider the flow over the newly added arcs in the transformed network to be the flow



a) The original network

b) The transformed network

---- Newly added arcs of infinite capacity

Fig. 1 Transformation of node-disjoint flow problem to arc-disjoint flow problem

through the node in the original network corresponding to that arc in the transformed network. Since the flows obtained are arc disjoint there is only one commodity flowing over the newly added arcs in the transformed network which makes the flow to be node disjoint in the original network. Since the newly added arcs are given infinite (a large number) capacity, nodes do not restrict the amount of flow. But solving the problem on the transformed network suffers from the drawback that the transformed network has more arcs and nodes than the original network. The problem of arc disjoint flow could also be solved using a branch and bound method similar to the one suggested in this work for arc disjoint flows. At each node of the branch and bound tree, instead of assigning an arc to a commodity, a node is assigned to a commodity. Obviously this will provide a solution to the problem with lesser amount of computation than the one provided by applying the solution procedure for arc

disjoint flow in the modified network, because the number of arcs and nodes in the original network is lesser than the number of arcs and nodes in the modified network.

The solution procedures for maximal arc disjoint flow problem in multicommodity common source network and maximal node disjoint flow problem in multicommodity common terminal (or common source) network are not discussed any further in this work as they are similar to the solution procedures for the maximal arc disjoint flow problem in multicommodity common terminal network.

CHAPTER 2

LITERATURE SURVEY

The class of network optimization problems, called the multicommodity flow problems arise in network modelling wherever commodities, vehicles or messages are to be shipped or transmitted from certain source nodes to certain terminal nodes of the underlying network. Recent applications of mathematical programming techniques to traffic equilibrium problems in transportation studies as well as computer networks analysis has generated considerable interest in this class of problems.

The multicommodity maximal flow problem arose out of the single commodity version. The efficiency of the labelling method for solving the single commodity maximal flow problem, and the theoretical insight offered by the Max-Flow-Min-Cut theorem led investigators to seek natural generalizations to the multicommodity version. Unfortunately such generalizations do not preserve the nice combinatorial features of the single commodity problem.

A multicommodity cut or disconnecting set is a set of arcs upon the removal of which no chains exist from the source to the terminal for any commodity. The multicommodity min-cut is one with minimal capacity. Because of weak duality, the maximal flow is always less than or equal to Minimal-cut and

the network is called "gapless" if equality holds. Single commodity problems are gapless by the Max-Flow-Min-Cut theorem, but the inequality can be strict for multicommodity flow problems. The multicommodity flow problem is not necessarily integer either. This means that the multicommodity flow problem may fail to have integral optimal solutions although arc capacities are integral. The integrality property for the single commodity problem followed from the total modularity of the constraint matrix. Jewell [6] has tried to extend this property to multicommodity flows but has met little success.

However, exploiting the combinatorial structure of the multicommodity problems, a few special results have been developed. Most of the special results are restricted to special networks or networks with atmost two sources or two or three commodities. Using matroid arguments, Evans et al. [2] show that a multicommodity transportation problem has a unimodular constraint matrix if and only if the number of sources or sinks is not greater than two.

The two commodity max flow problems has been investigated extensively. Algorithms for finding the maximal two-commodity flow on undirected graphs are given in Hu [5]. Hu shows that this problems has integral solution for even arc capacities. It is hard to extend these results to problems with three or more commodities except in very special cases. For three-commodity flow problem, Rothfarb et al [12] prove that a special three-

commodity undirected graph in which all nodes are either sources or sinks is gapless and give an algorithm to find the maximal flow.

Algorithms are available for other specialized maximal flow problems. Kleitman [8] solves the case where each node is a sink for all commodities but one. Rothfarab et al [11] consider the multicommodity network flow problem where all the commodities are destined for a common terminal. They have presented an algorithm to maximize $\sum_{k=1}^{K} k\, F^k$ for decreasing weights $\alpha > \alpha^2 > \ldots > \alpha^k$, where K is the number of commodities.

Ford and Fulkerson [4] have suggested solution techniques to the multicommodity maximal flow problems. They proposed the arc-chain formulation of the problem, and solved the problems using revised simplex method.

The problem of arc disjoint flow in multicommodity network was studied by Ahuja [1] . He suggested a branch and bound algorithm for the solution of this problem. The upper bound was obtained by solving p independent single commodity maximal flow problems where p is the number of commodities.

A thorough survey of the literature suggests that practically no work has been published in the area of arc-disjoint flow in multicommodity networks with a common terminal node. This acted as the prime motivation for the present work.

CHAPTER 3

MATHEMATICAL FORMULATION

Most of the network flow problems lend themselves in a natural manner to Linear Programming Formulation. Therefore, it is possible to solve these problems using the available techniques for the solution of linear programming problems. However, due to the special structure of the network flow problems number of efficient algorithms and elegant theorems have been developed. In most of the network flow problems, the optimum solutions are always integers, a fact which is not true in linear programs. But this property of the single commodity flow that the optimal solutions are integers is not always true for multicommodity networks. Yet it is to be noted that the multicommodity flow problem taken up in this work has integral solutions. This is clear from the fact that the solution to the problem is the sum of the solution of p single commodity single source, single terminal flow problems (p is the number of commodities to be flown in the network) solved over p networks, each of which is made up of mutually exclusive set of arcs. Solutions for each of these individual problems are integral and so the sum of p integral solution will also be integral. Exploiting this special property of the linear program solution, we could develop better solution procedures than the linear program formulation of the problem.

The formulation of the problem as a mixed integer linear program is given below.

The network $G(N,A)$ is a finite set of nodes N containing n elements and a finite set of ordered pair of nodes called arcs, A containing m elements. We shall use N_i to indicate the **ith node** and A_{ij} to indicate the directed arc leading from N_i to N_j. A network is connected if for every partitioning of nodes, of the network into subsets X and $\bar{X}$, there is either an arc A_{ij} or A_{ji} with $N_i \in X$ and $N_j \in \bar{X}$. The word 'network' will be used to denote connected network throughout this work. It is also assumed that all the arcs are directed and there is atmost one directed arc between any pair of nodes. That is both A_{ij} and A_{ji} will not be members of the set of arcs. Clearly, the problem of maximum flow in a network is meaningful only if the arcs of the network or a subset of it possess upper limits called the capacities of the arc. Let each arc A_{ij} have a capacity b_{ij} which is the maximum amount that can flow over that arc.

In the multicommodity network considered, more than one distinct commodity flow over the network with S_i as the originating node (source node) for the ith commodity and 't' as the destination node (terminal node) for all the commodities. Let p be the number of commodities that are to flow over the network. So there will be p source nodes and one common terminal node in the network.

Associated with each node of the network, there are two integer numbers called in-degree and out-degree. In-degree of a node is the total number of arcs coming into that node and the out-degree is the total number of arcs going out of that node. It is also assumed that the in-degree of all the source nodes and the out-degree of the common terminal node are zero.

The problem can be posed as a mixed integer linear programming problem.

Let v^k denote the flow of the kth commodity between the source node for that commodity and the common terminal. Let x_{ij}^k denote the amount of flow of the kth commodity over the arc A_{ij}.

Let us introduce the zero-one variables y_{ij}^k where

$y_{ij}^k = 1$ if the kth commodity passes over arc A_{ij}

$= 0$ otherwise.

There will be mxp variables of the type x_{ij}^k and another mxp variables of the type y_{ij}^k where m and p are the number of arcs and the number of commodities respectively.

Objective function :

The objective is to maximize the sum of the flows of all commodities. So the objective function will be

$$\text{Maximize} \quad \sum_{k=1}^{p} v^k$$

Constraints

The constraints of this problem are of 4 types. They are given below.

1. Flow conservation constraints :

These constraints represent the fact that for each commodity, the total flow emerging out of its source node and the total flow terminating in the common terminal node are numerically equal to the total flow of that commodity and for all nodes other than the source nodes and the common terminal node, the flow is conserved (i.e. the total incoming flow is equal to the total out-going flow).

These constraints can be expressed mathematically as

$$\sum_{j=1}^{n} x_{ij}^{k} - \sum_{j=1}^{n} x_{ji}^{k} = \begin{cases} v^{k} & \text{if } i = S_k \\ -v^{k} & \text{if } i = t \\ 0 & \text{if } i \text{ is any node other than the source nodes and the common terminal node.} \end{cases}$$

$$\text{for } k = 1, 2, \ldots p.$$

Corresponding to each node of the network, there are p constraints of this type. Thus the total number of flow conservation constraints is nxp.

2. Arc disjoint constraints :

These constraints represent the restriction that more than one commodity should not flow over an arc. This restriction can be expressed mathematically as

$$\sum_{k=1}^{p} y_{ij}^{k} \leq 1 \qquad \text{for all}(i,j) \in A,$$

On the arc A_{ij}, flow of the kth commodity can be positive only if the value of the variable y_{ij}^{k} is 1. For any other commodity k' (i.e. k' = 1,2, ... p, $\neq$ k) because of the above constraint, the corresponding zero-one variables $y_{ij}^{k'}$ takes zero value; thus indicating that there is no flow of the commodity k' over the arc A_{ij}.

For each arc in the network, there is a corresponding arc disjoint constraint. Thus the total number of arc disjoint constraints is m.

3. Capacity constraints :

These constraints ensure that the total flow over an arc will not be more than the capacity of that arc.

Since the arc disjoint constraints take care that only one commodity flows over an arc, the capacity constraints have to only restrict the flow of any commodity over the arc A_{ij} to be not greater than its capacity b_{ij}. (It is not necessary to restrict the sum of the flows of different commodities

over the arc to be not greater than the capacity of the arc).

The mathematical expression of these constraints is

$$x_{ij}^k \leq b_{ij} \times y_{ij}^k \qquad \text{for all } (i,j) \in A$$

$$k = 1,2, \ldots, p.$$

Corresponding to each arc in the network, there will be p such constraints. Therefore, the total number of capacity constraints are mxp.

4. Non-negativity constraints :

The non-negativity constraints restrict the flow of any commodity over an arc to be non-negative. Thus,

$$x_{ij}^k \geq 0 \qquad \text{for all } (i,j) \in A$$

$$k = 1,2, \ldots, p$$

There are mxp constraints of this type.

Also

$$y_{ij}^k = 0 \text{ or } 1.$$

There are another mxp constraints of this type too.

As seen in the formulation, this is a mixed integer linear programming problem where the variables y_{ij}^k are restricted to 0 or 1. For a network with m arcs, n nodes and p commodities, the total number of variables in the mathematical formulation will be 2xmxp. Out of this mxp are zero-one variables.

There will be (3xmxp) + (nxp) + n constraints of which mxp constraints are the constraints which restrict x_{ij}^{k} to be non-negative and another mxp constraints are the constraints which restrict y_{ij}^{k} to take values either 0 or 1.

The problem can be solved by any one of the mixed integer programming algorithms. However, the solution of a moderately sized problem by such an approach may be prohibitive because of large computer memory requirements and computational complexity.

CHAPTER 4

SOLUTION METHODOLOGY

In this chapter, two algorithms are developed for solving the maximal arc disjoint flow problem in a multi-commodity common terminal network. Both the algorithms are based on the well known branch and bound strategy. The branch and bound approach is a very useful tool for solving combinatorial optimization problems. In the process of seeking the optimal solution, it implicitly enumerates all possible solutions. It performs it systematically and aims to do it with least computations.

4.1 Branch and Bound Philosophy

In the course of applying the branch and bound approach, the overall set of feasible solutions to the problem, is partitioned into many simpler subsets. At each stage, one promising subset is chosen and an effort is made to find the best feasible solution from it. If it is found, then that subset is said to be fathomed. If the best feasible solution is not found, then that subset is again partitioned into two or more simpler subsets (this operation is known as branching) and the same process is repeated again. At each partitioning of a set into its subsets, a lower bound (or upper bound) which gives the minimum value (or maximum value) of the

objective function which any feasible solution in that subset may yield. If this bound is greater than (or lower than) some best known feasible solution (known as incumbent) then that subset is not further branched. This operation is known as pruning. When all the subsets are either fathomed or pruned, incumbent provides the value of optimal solution.

4.2 Algorithm - 1

4.2.1 Upper bounding strategy

Let us consider a network through which p commodities are to flow. At any node of the branch and bound tree developed for the network, the arcs of the network can be dividied into (p+1) groups. Let A_o be the set of arcs which are assigned to no commodity. Further let A_1, A_2, A_3 , ... A_p represent the set of arcs assigned to commodities 1,2,3, ... p, respectively.

It is to be noted that the sets A_1, A_2, ... A_p, A_o are mutually exclusive and collectively exhaustive sets.

To find the upper bound for each node of the branch and bound tree, a super source is added to the network and each of the source nodes are connected to this super source with arcs of infinite (large) capacity. The maximal flow from the super source to the common terminal which could be found out by a modified maximal flow labelling algorithm in

which commodity 'i' is allowed to pass only through the set of arcs $A_i \cup A_o$, is used as the upper bound flow for the subproblem corresponding to that node of the branch and bound tree.

It is to be noted that for finding the upper bound of any subproblem, we are not solving p, single commodity single source, single terminal flows independently which would involve the following procedure.

For commodity 'i', remove all the arcs which are assigned to other commodities from the network and find the maximal flow from the source of the ith commodity to common-terminal. The same process is repeated for all values of 'i'. The sum of the p flow values thus obtained will logically form an upper bound for the problem. But in this case, both the capacity constraints and arc disjoint constraints are violated and so will provide only a loose upper bound for the problem.

The upper bound obtained by the proposed strategy will be close to the solution, as in the flow corresponding to this upper bound, only the arc disjoint constraints are violated for the arcs in the set A_o, but the capacity constraints are maintained for all arcs. This tightness of the upper bound accounts for the efficiency of the branch and bound algorithm suggested.

Logically the upper bound value decreases in further assignments, as at each stage an unassigned arc get assigned to one of the commodities. So its flexibility of getting assigned to any of the commodities is lost.

The proof for the validity of this upper bound for the maximal flow for any particular assignment (i.e. at any particular node of the branch and bound tree) is given below.

Proof :

Obviously the labelling algorithm terminates when no augmenting path exists between the source of ith commodity and the common terminal over the set of arcs $A_i \cup A_o$ for all values of 'i'. Let v be the total flow value of such a flow obtained.

Let there exists another flow for the same assignment of arcs for which the total flow value is v_1 and $v_1 > v$.

This means that for at least one of the commodities, the flow from its source to the common terminal should be more than the present flow. But, since the algorithm terminates only when no augmenting path is possible from any source to the terminal, an increased flow can be achieved only if the flow of some other commodities are reduced over arcs with which these commodities share at least one arc. Very obviously, these arcs cannot be in the sets $A_1 \ldots A_p$ as these arcs allow the flow of only the commodity to which it

is assigned. So to get an increased flow of some commodities from their source nodes to the common terminal node, flow of those commodities over the arcs in set Δ_0 are to be increased and flow of other commodities over the arcs in set Δ_0 are to be decreased and the increment in flow obtained should be greater than the decrement of flow of other commodities.

Since all commodities are allowed to flow over the arcs in set Δ_0, if any such augmenting paths had been possible, the labelling algorithm would have detected it before termination. At termination of the labelling algorithm no such augmentations are possible and so there can not exist a flow for which the total flow value, v_1 is greater than v, for the same assignment of arcs to different commodities.

4.2.2 Branching Strategy

Branching is done at each node of the branch and bound tree, considering the flow over all the arcs in the solution of the maximal flow problem at that node. Of all the arcs which allows simultaneous flow of more than one commodity, the arc which has the maximum total flow (i.e., the sum of flows of all commodities) is choosen for branching). The arc thus selected serves as the basis of branching at that node and as many new nodes are created in the branch and bound tree as the number of commodities. For each node thus created, the arc selected for branching is assigned

seperately to each of the commodities.

After each branching, the newly generated nodes of the branch and bound tree are examined and the feasible solutions are used for updating the incumbent.

4.2.3 Searching Strategy

The searching strategy selects the node of the branch and bound tree at which branching is done. Of the nodes which are added latest to branch and bound tree, the one which has the largest upper bound is choosen for further branching. This results in what is commonly termed as 'depth first search strategy'. In this method, instead of scanning every node which is directly connected to (adjacent to) the current node, we move to one adjacent node, as soon as possible,leaving the current node with possibly unexplored adjacent nodes for the time being. In other words, we trace a walk through the branch and bound tree, going on to a new node whenever possible.

4.2.4 Notations Used :

A	-	set of arcs in the network
p	-	number of commodities to be flown in the network
S_i	-	source of the ith commodity
S	-	super source added to the network
T	-	common terminal node in the network
X_{ij}^k	-	flow of the kth commodity over arc (i-j)

A_i - The set of arcs assigned to the ith commodity at any node of the branch and bound tree

A_o - the set of arcs which are assigned to none of the commodities at any node of the branch and bound tree

L - $\{P_i\}$ - set of active infeasible subproblems (p_i) which form the unfathomed nodes of the branch and bound tree

I - value of the current incumbent

$U(p_i)$ - upper bound of the subproblem p_i

4.2.5 Step-wise Description of the Algorithm

Step - 0 :

Add a super source S and connect each source node S_i to this super source with arcs of infinite capacity.

Set I = 0

Set L = {Ø}

Set A_o = {All arcs in the network}

Set A_i = { Ø } for i = 1,2, ... p.

Set X_{ij}^k = 0 for all $(i,j) \in A$,

$k = 1,2, \ldots p.$

Solve the maximal flow problem from super source S to common terminal T. If the flow values are feasible, i.e. no arc has more than one commodity flowing over it; STOP (This is the optimum flow). Otherwise, let p_o represent this subproblem. Add p_o to the list L.

Step - 1 :

Check List L. If L = $\{\emptyset\}$, go to Step 6. Otherwise, take out the problem which has the largest upper bound of all the problems which are added latest to the branch and bound tree. Let it be p_m. Update List L.

Step - 2 :

If $U(p_m) < I$, then prune that node of the branch and bound tree and go to Step - 1. Otherwise scan all $(i,j) \in A$. Determine the arc which flow more than one commodity over it and has maximum $\sum_{k=1}^{p} x_{ij}^k$, among all such arcs.

Step - 3 :

Generate new subproblems (new nodes for branch and bound tree), p_{m1}, p_{m2}, ... p_{mp}. Subproblem p_{mi} has the arc selected for branching assigned to commodity 'i'.

Step - 4 :

Solve the subproblems p_{m1}, p_{m2}, ... p_{mp} (the subproblems corresponding to the newly generated nodes of the branch and bound tree) for maximal flow. Commodity 'i' is allowed to flow only over the set of arcs $A_i \cup A_o$. The maximal flow thus obtained for the subproblem p_{mi} is used as $U(p_{mi})$. Check whether the problem has feasible flow. If so; go to Step 5.

Add all the subproblems with infeasible solutions to the List L in the increasing order of their corresponding upper bounds $U(p_{mi})$. Go to Step - 1.

Step - 5 :

If $U(p_{mi}) > I$, set $I = U(p_{mi})$. Go to Step - 4.

Step - 6 :

Current incumbent is the maximal flow value. Stop.

4.2.6 Example

The suggested procedure is illustrated with the help of a small network shown in Fig. 2. The three commodity network considered has 10 nodes and 12 arcs. All the source nodes are connected to a super source with fictitious arcs of infinite (a large number) capacity shown by dotted lines. The circled numbers placed near the end of the arc is the arc number and the uncircled number placed near the beginning of the arc is the capacity of that arc.

The working of the algorithm is described step-wise.

Step - 0 :

Set $I = 0$ and L to be empty. All arcs are included in the set A_0. A_1, A_2 and A_3 are set to be empty. The maximal flow from the super source to common terminal found out using labelling algorithm is 25. Flow values are infeasible. Let the current assignment of arcs correspond

to subproblem (1) with upper bound value of 25. This problem is added to List L.

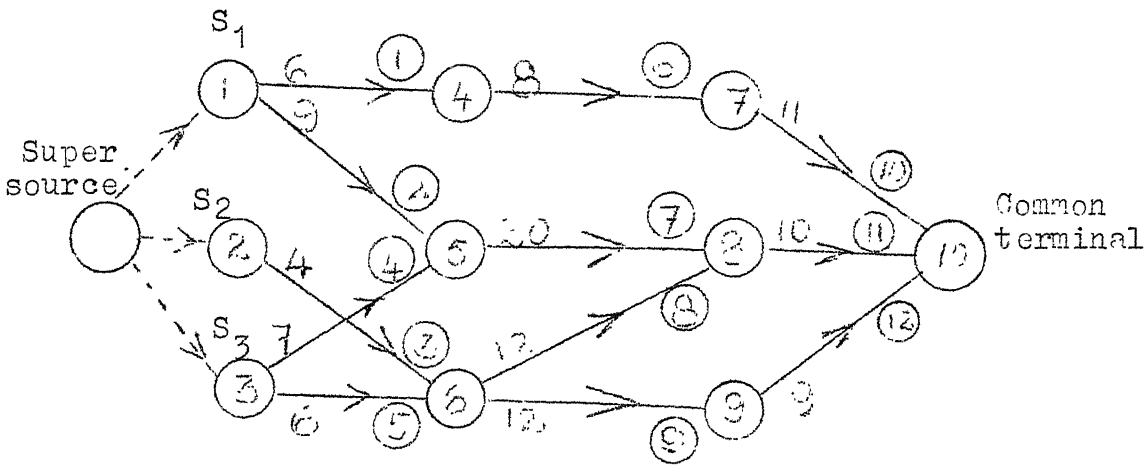


Fig. 2 An illustrative example

Step - 1 :

The List L is checked. It contains one subproblem and it is taken out. List L is updated.

Step - 2 :

Since upper bound of the subproblem is greater than the incumbent, all arcs are scanned and arc 11 is choosen as the basis of branching because it is the arc which allows the maximum total flow of all the arcs which allow the flow of more than one commodity.

Step - 3 :

The nodes 2,3 and 4 are generated in the branch and bound tree. Nodes 2,3 and 4 correspond to subproblems in which arc 11 is assigned to commodities 1,2 and 3 respectively.

Step - 4 :

Solving the subproblems corresponding to nodes 2,3 and 4, it is found that U(2) = 24, U(3) = 16, U(4) = 23. Subproblem corresponding to node 3 has feasible solution. Therefore, go to Step - 5.

Step - 5 :

Since U(2) > I, I is set to U(2) = 16. Go to Step - 4.

Step - 4 :

The infeasible subproblems are added to List L in the increasing order of their upper bounds. Thus subproblems corresponding node 2 becomes the last element in List L. Go to Step 1.

Iteration - 2 :

Step - 1 :

The last element of List L (i.e. the subproblem corresponding to node 2) is taken out from List L.

Step - 2 :

Since the upper bound of this subproblem is greater than the incumbent, scan all the arcs and arc number 12 is choosen to be the basis for further branching.

Step - 3

New nodes 5,6 and 7 are created in the branch and bound tree.

Step - 4 :

The solution of the subproblems yielded the following values for upper bound U(5) = 15, U(6) = 19, U(7) = 21. Since all the problems have feasible flow, go to Step - 5.

Step - 5 :

Since U(5) < I, node 5 is pruned. Since U(6) > I, I is set to U(6) = 19. Since U(7) > I, I is set to U(7) = 21. Go to Step - 4.

Step - 4 :

Since none of the subproblems corresponding to the newly generated branch and bound nodes are infeasible, nothing is added to the List L.

Iteration - 3 :

Step - 1 :

Last element in List L (i.e. the subproblem corresponding to node 4 of the branch and bound tree) is taken out.

Step - 2 :

Upper bound of this subproblem is greater than incumbent. Arc 12 is selected as a basis for branching.

Step - 3 :

3 new nodes 8,9 and 10 are created in the branch and bound tree.

Step - 4 :

The subproblems corresponding to the newly generated branch and bound nodes are solved. U(8) = 16, U(9) = 20, U(10) = 19. Since all these subproblems have feasible solutions, go to Step - 5.

Step - 5 :

The upper bound for all the three subproblems are less than the current incumbent and so all those nodes are pruned. Go to Step - 4.

Step - 4 :

None of the newly generated subproblems gave infeasible solutions. So nothing is added to List L. Go to Step 1.

Iteration - 4 :

Checked the list L and it is found empty. Stop.

The branch and bound tree developed by the algorithm for this problem is shown in Fig. 3.

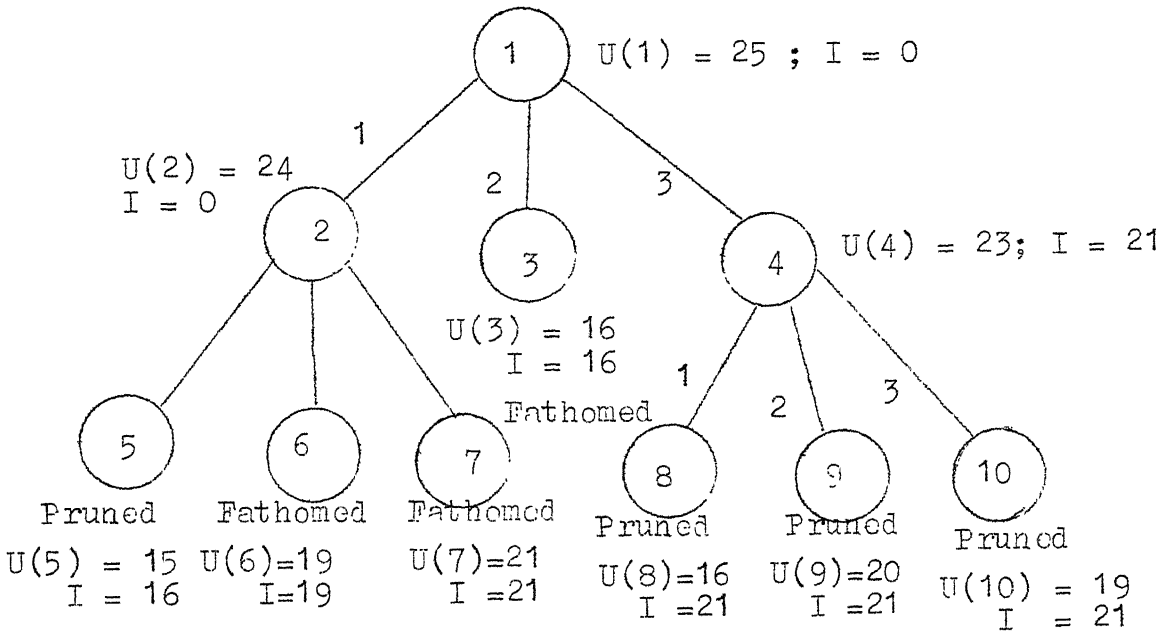


Fig. 3 Branch and bound tree generated for the example problem

4.3 Algorithm - 2

This algorithm is also based on branch and bound strategy and is similar to the algorithm suggested earlier. The branching and searching strategies used are the same as those used in the previous algorithm. The basic difference lies in the procedure used for finding the upper bound for any node in the branch and bound tree, Instead of starting with an initial flow of zero over all the arcs, the maximal flow in the problem corresponding to the node from which the node under consideration originated is made use of. Here also each node of the branch and bound tree represents a unique assignment of arcs to different commodities. That means, that at each node of the branch and bound tree, the arcs of

the network can be divided into mutually exclusive and collectively exhaustive sets of arcs A_o, A_1, ... A_p where A_o is the set of arcs which are not assigned to any commodity and A_i is the set of arcs which are assigned to the ith commodity. The upper bound will be the maximal flow from the source nodes to the common terminal node considering the commodity assignments done on arcs. For obtaining the maximal flow, we employ the same modified labelling algorithm which we used in the previous algorithm, the validity of which is proved already.

4.3.1 Upper Bounding Strategy

A super source is added to the network and this super source is connected to the source nodes by arcs of infinite capacity. For finding the upper bound for any branch and bound node, the maximal flow found out for calculating the upper bound for the node from which the node under consideration is generated, is used. From this flow, the flow paths containing the arc selected for branching are detected. These paths are removed from the flow. After this is done, the flow of all the commodities over arc selected for branching will be zero.

All the arcs connecting the super source with the sources other than the source for the commodity to which the selected arc is assigned at the node under consideration are removed from the network. Using the modified labelling

algorithm, in which the flow of the ith commodity is allowed only over the arcs in the set $A_o \cup A_i$, we find the maximal flow from the super source to the common terminal. Next, the arc connecting the super source with the source for the commodity to which the candidate arc is assigned, and the candidate arc are removed from the network. All the other arcs connecting the super source with the source nodes are put back in the network. Keeping present flow as the starting flow the maximal flow from the super source to the common terminal on this network is found out using the modified maximum flow labelling algorithm. This gives the upper bound for the subproblem (node of the branch and bound tree, considered).

Proof :

Obviously, the flow got after removing all the flow paths which contain the candidate arc is a feasible and sub-optimal solution to the subproblem under consideration. Since, in this subproblem, the candidate arc is assigned to only one commodity, when we apply the maximal flow algorithm, in the network in which all source nodes other than the source node of the commodity to which the candidate arc is assigned, are disconnected from the super source, we get another feasible solution to the subproblem under consideration which contains the maximal flow of that commodity to which the candidate arc is assigned. The possible incremental flows of all other

commodities are obtained by again applying the maximal flow algorithm on the network containing all arcs of the original network other than the candidate arc and the arc connecting the super source with the source of the commodity to which the candidate arc is assigned. Since no incremental flow is possible for any commodity, this will give the maximal total flow of all commodities to the common terminal, for any particular assignment of arcs to different commodities.

4.3.2 Description of the Algorithm

The notations used here are the same as the notations used in the description of Algorithm - 1.

Step - 0 :

Add a super source S and connect each source S_i to this super source with arcs of infinite capacity. Set

$I = 0$

$L = \{\emptyset\}$

$A_o = \{$All arcs in the network $\}$

$A_i = \{\emptyset\}$ for $i = 1, \ldots p$, and

$x^k_{ij} = 0$ for all $(i,j) \in A$, $k = 1,2, \ldots, p$.

Solve the maximal flow problem from super source S to common terminal T. If the flow values are feasible, i.e., no arc has more than one commodity flowing over it; STOP (This is the optimal flow). Otherwise let p_o represent the sub-problem. Add p_o to the List L. Go to Step - 1.

Step - 1 :

Check List L. If L = {$\emptyset$} , go to Step - 6. Otherwise, take out the problem which has the largest upper bound of all the problems which are added latest to the branch and bound tree. Let it be p_m. Update List L, and go to Step - 2.

Step - 2 :

If $U(p_m) \leq I$, then prune that node of the branch and bound tree and go to Step - 1. Otherwise scan all $(i,j) \in A$, determine the arc which flows more than one commodity over it and has the maximum $\sum_{k=1}^{p} x_{ij}^k$ among all such arcs. Go to Step - 3.

Step - 3 :

Generate new subproblems (new nodes for branch and bound tree) p_{m1}, p_{m2}, ... p_{mp}. Subproblem, p_{mi} has the arc selected for branching assigned to commodity 'i'. Go to Step - 4.

Step - 4 :

i) Solve the subproblem p_{m1}, p_{m2}, ... p_{mp} (The subproblems corresponding to the newly generated nodes of the branch and bound tree).

ii) Trace and remove all the flow paths which contain the arc selected for branching in the subproblem p_m (pushout routine).

iii) Remove the arcs connecting the super source with the source nodes of all commodities other than commodity 'i'. Using the present flow as the initial flow, find the maximal flow from the super source to common terminal. Commodity 'j' is allowed to flow only over the set of arcs $A_j \cup A_o$.

iv) Now put all the arcs removed, back into the network. Remove the arc connecting the super source with the source node of the commodity 'i', and the arc selected for branching in the subproblems p_m. Using the present flow as the initial flow, find the maximal flow from the super source to the common terminal node, considering the assignment of arcs to different commodities. The maximal flow thus obtained for the subproblem p_{mi} is used as $U(p_{mi})$.

Check whether the problem has a feasible flow. If so go to Step - 5.

v) Add all the subproblems with infeasible solution to the list L in the increasing order of their corresponding upper bound $U(p_{mi})$. Go to Step - 1.

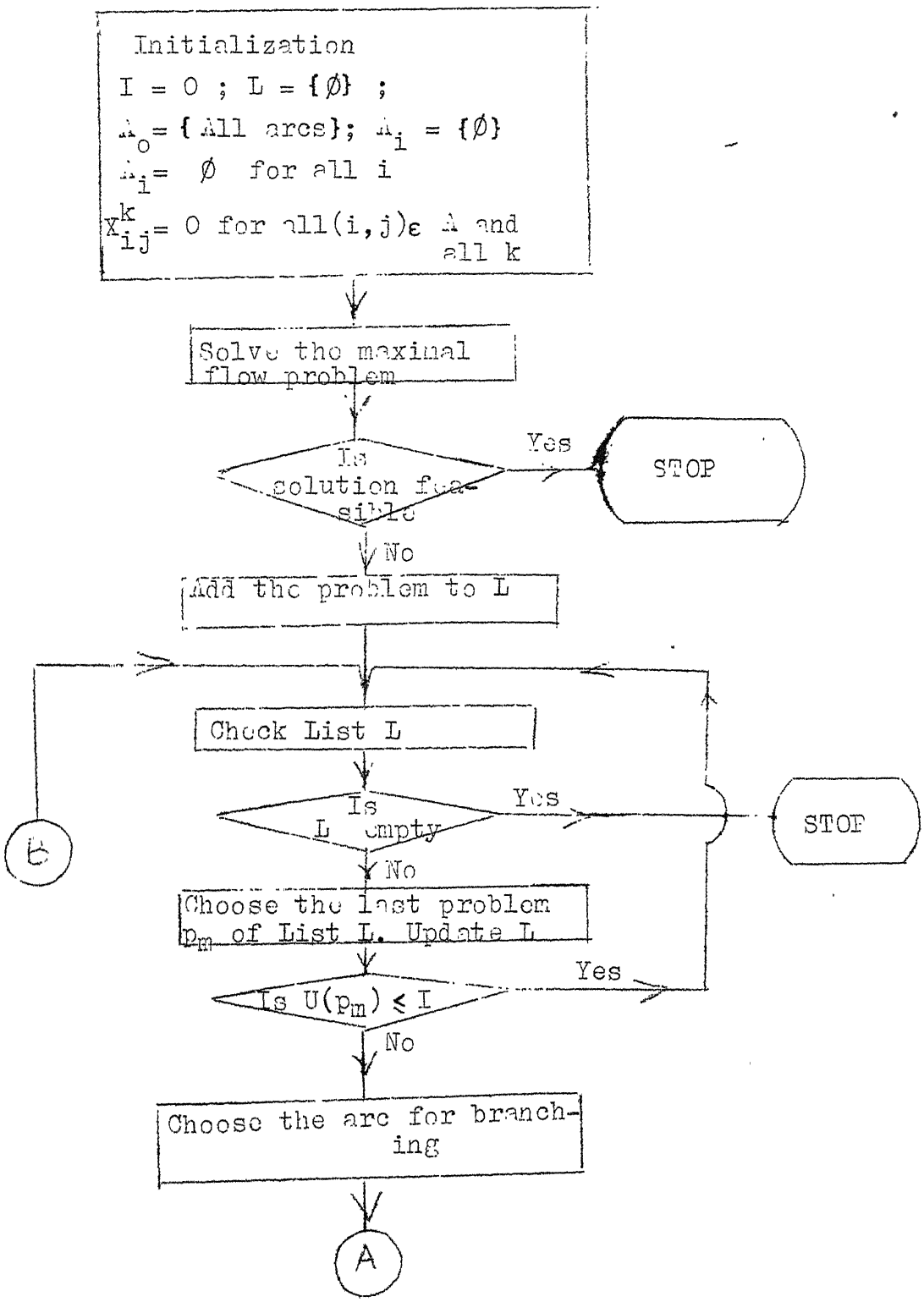
Step - 5 :

If $U(p_{mi}) > I$, set $I = U(p_{mi})$. Go to Step - 4.

Step - 6 :

Current incumbent is the maximal flow value. Stop.

A general flow chart for both the proposed branch and bound algorithm is given in Fig. 4.


Initialization
I = 0 ; L = {Ø} ;
A_o = { All arcs}; A_i = {Ø}
A_i = Ø for all i
X^k_{ij} = 0 for all (i,j)ε A and all k
Solve the maximal flow problem
Is solution feasible
Yes
STOP
No
Add the problem to L
Check List L
Is L empty
Yes
STOP
No
Choose the last problem p_m of List L. Update L
Is U(p_m) ≤ I
Yes
No
Choose the arc for branching
B
A

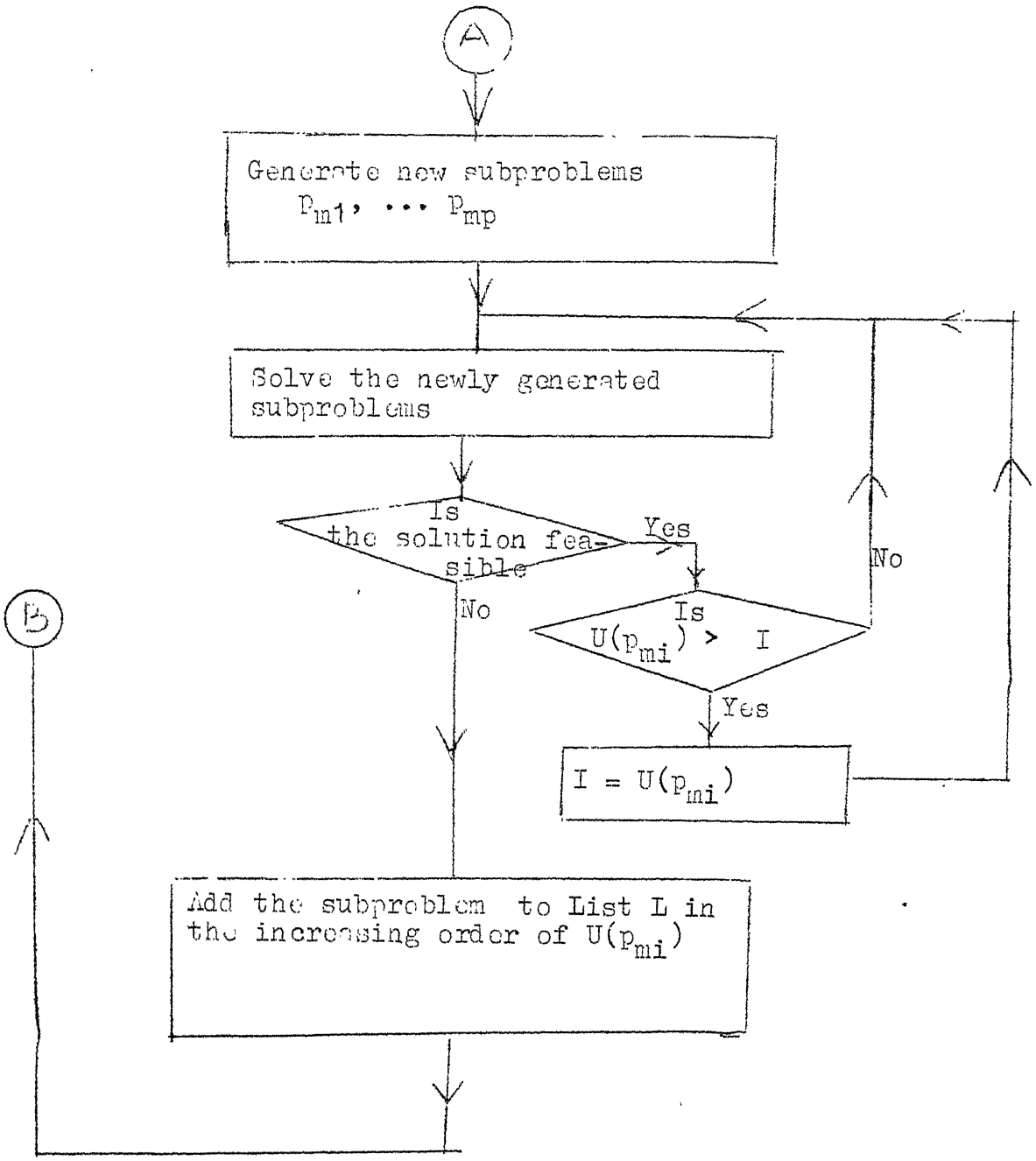


Fig. 4 A general logic flow diagram for the proposed algorithms

CHAPTER 5

COMPUTATIONAL PERFORMANCE EVALUATION OF ALGORITHMS AND CONCLUSIONS

5.1 Computational Performance Evaluation

The algorithms developed were coded in FORTRAN-10 and implemented on DEC-1090 time-sharing computer system. The programs were tested over a large number of networks generated randomly.

The generation of the network is done as follows :

The node-node incidence matrix is generated randomly. In the generation of the incidence matrix, it is taken care that the graph corresponding to this incidence matrix is connected and there would be atmost one directed arc between any pair of nodes. It was also taken care that all the source nodes have zero indegree and the terminal node has zero out-degree. The parameters of the randomly generated network are the number of nodes, number of arcs and the number of commodities.

The computational performance of the proposed methods are evaluated on the basis of the number of branch and bound nodes created and the CPU (Central Processing Unit) time. The maximum number of branch and bound nodes that could be created is $1 + p + p^2 + \ldots + p^m$. But in most of

the branch and bound algorithms only a fraction of this is actually created before termination. Lesser the number of branch and bound nodes created the efficient the procedure is.

In DEC-1090 system which is a multiprogramming system, the program remains in a state of execution for a relatively long time while the central processing unit takes it up only intermittently.

For each combination of the parameters of the network five problems were generated and programs for algorithm 1 and algorithm 2 were tested on these problems. Problems upto the size of 200 nodes 300 arcs and 5 commodities were solved. The computational experience is presented in Tables 1 to 4.

The number of feasible solutions to the problem taken up in this work is proportional to p^m where p is the number of commodities and m is the number of arcs. So as the number of arcs and the number of commodities increase, the computational complexity also increase. Typically in a branch and bound procedure only a fraction of the feasible solutions are enumerated. In the suggested algorithms, because of the tight upper bound, the number of feasible solutions enumerated explicitly is expected to be very small and so moderately sized problems could be worked out using these algorithms. Most of the problems tested were solved in less than 5 seconds of C.P.U. time.

Table 1 The computational experience on two commodity networks of different sizes

No. of nodes	Density: Less than 0.1			0.1 - 0.3			0.3 - 0.5			Above 0.5		
	NN	T1	T2	NN	T1	T2	NN	T1	T2	NN	T1	T2
10	-	-	-	-	-	-	0.8	443	594	0.8	637	735
20	-	-	-	0.4	365	546	1.5	932	1376	1.2	1034	1254
30	0.2	19	41	0.6	384	682	2.3	1436	1943	3.1	2546	2795
40	0.4	37	59	0.6	437	722	3.1	2371	2896	-	-	-
50	0.4	64	186	1.8	637	848	-	-	-	-	-	-
60	0.8	96	220	2.6	893	1037	-	-	-	-	-	-
70	0.8	143	334	2.8	976	1236	-	--	-	-	-	-
80	0.8	166	467	-	-	-	-	-	-	-	-	-
90	1.0	228	512	-	-	-	-	-	-	-	-	-
100	1.0	334	664	-	-	-	-	-	-	-	-	-
Above 100	1.0	512	756	-	-	-	-	-	-	-	-	-

NN - Average Number of nodes created in the branch and bound tree

T1 - Average CPU time (in milliseconds) taken for algorithm - 1

T2 - Average CPU time (in milliseconds) taken for algorithm - 2

Table 2 The computational experience on three commodity networks of different sizes

Density / No. of nodes	Less than 0.1			0.1 - 0.3			0.3 - 0.5			Above 0.5		
	NN	T1	T2	NN	T1	T2	NN	T1	T2	NN	T1	T2
10	-	-	-	-	-	-	0.6	134	236	1.8	243	314
20	-	-	-	0.8	264	314	0.9	283	547	4.3	662	834
30	0.2	15	41	1.7	448	665	2.7	612	1037	5.8	1676	1894
40	0.3	28	83	2.3	676	981	3.5	836	1434	-	-	-
50	0.4	76	158	2.3	838	1438	-	-	-	-	-	-
60	0.9	142	326	3.1	984	1746	-	-	-	-	-	-
70	1.1	183	391	3.7	1134	2306	-	-	-	-	-	-
80	1.1	264	674	-	-	-	-	-	-	-	-	-
90	1.2	296	737	-	-	-	-	-	-	-	-	-
100	1.2	412	793	-	-	-	-	-	-	-	-	-
Above 100	1.2	634	984	-	-	-	-	-	-	-	-	-

NN - Average number of nodes created in the branch and bound tree

T1 - Average CPU time (in milliseconds) taken for algorithm - 1

T2 - Average CPU time (in milliseconds) taken for algorithm - 2

Table 3 The computational experience on four commodity networks of different sizes

No. of nodes	Density: Less than 0.1			0.1 - 0.3			0.3 - 0.5			Above 0.5		
	NN	T1	T2	NN	T1	T2	NN	T1	T2	NN	T1	T2
10	-	-	-	-	-	-	0.8	474	867	1.8	1234	1683
20	-	-	-	0.8	290	456	1.5	904	1467	4.4	1695	2347
30	0.3	67	86	2.1	464	737	2.5	1534	2235	5.8	2836	2967
40	0.4	104	237	2.5	638	963	3.0	2114	2637	-	-	-
50	0.4	136	330	2.6	834	1234	-	-	-	-	-	-
60	0.7	187	465	3.6	1038	1567	-	-	-	-	-	-
70	1.2	234	570	4.2	1432	2604	-	-	-	-	-	-
80	1.3	267	584	-	-	-	-	-	-	-	-	-
90	1.3	344	633	-	-	-	-	-	-	-	-	-
100	1.3	467	796	-	-	-	-	-	-	-	-	-
Above 100	1.4	640	1034	-	-	-	-	-	-	-	-	-

NN - Average number of nodes created in the branch and bound tree

T1 - Average CPU time (in milliseconds) taken for algorithm - 1

T2 - Average CPU time (in milliseconds) taken for algorithm - 2

Table 4 The computational experience on five commodity networks of different sizes

No.of nodes	Density: Less than 0.1 NN	T1	T2	0.1 - 0.3 NN	T1	T2	0.3 - 0.5 NN	T1	T2	Above 0.5 NN	T1	T2
10	-	-	-	-	-	-	1.4	1037	1654	2.7	1731	2016
20	-	-	-	1.8	476	1076	2.1	1834	3211	6.3	3143	4015
30	0.5	136	377	3.8	839	2011	3.9	3014	3761	9.7	4038	4817
40	0.8	249	482	4.1	1534	2512	6.3	3617	4634	-	-	-
50	0.8	368	734	5.6	2611	3778	-	-	-	-	-	-
60	1.3	437	812	6.3	2870	4030	-	-	-	-	-	-
70	1.5	488	890	6.5	3046	4994	-	-	-	-	-	-
80	1.5	531	1136	-	-	-	-	-	-	-	-	-
90	1.7	607	1768	-	-	-	-	-	-	-	-	-
100	1.8	676	2241	-	-	-	-	-	-	-	-	-
Above 100	1.8	896	2784	-	-	-	-	-	-	-	-	-

NN - Average number of nodes created in the branch and bound tree

T1 - Average CPU time (in milliseconds) taken for algorithm - 1

T2 - Average CPU time (in milliseconds) taken for algorithm - 2

Algorithm - 1 was found to be computationally superior to Algorithm - 2 as it has less memory requirements and also takes less C.P.U. time. In almost all problems Algorithm - 2 took much more time than Algorithm - 1. But it was noticed that as the density of the graph increased, the difference in C.P.U. time between Algorithm - 1 and Algorithm - 2 decreased.

5.2 Conclusions

For the problem of finding the maximal arc disjoint flow in a multicommodity network considered in this work, two algorithms are proposed. Both the algorithms are based on the branch and bound philosophy. As the upper bounding strategy employed in both the algorithms violates only one constraint (i.e. the arc disjoint constraint) of the original problem, tight upper bound is obtained. This explains for the low number of nodes created in the branch and bound trees generated by these algorithms and their high computational efficiency. These conclusions are based on the experimentation conducted on a large number of randomly generated networks. In almost all the networks generated for experimentation, Algorithm - 1 was found to be computationally more efficient as it took less C.P.U. time and required less core memory than Algorithm - 2.

5.3 Scope for Future Work

The solution procedures proposed for the class of disjoint flows considered in this work are based on the branch and bound technique and so implicit enumeration of all the feasible solutions are done. It may be worthwhile to think in terms of exploiting the combinatorial structure of the problem and come up with more elegant methods like labelling method.

The present work was concerned with pure disjoint flows, i.e., where all arcs or nodes allow only one commodity to flow over it. Further work could be done by considering the mixed disjoint flows, i.e., where some arcs or nodes allow simultaneous flow of commodities.

REFERENCES

1. AHUJA, R.K., "Maximal Arc-disjoint and Node-disjoint Flow in a Multicommodity Network", Unpublished M.Tech. Dissertation, Industrial & Management Engineering Program, Indian Institute of Technology, Kanpur, July, 1979.

2. EVANS, J.R., J.J. Jarvis and R.A. Duke, "Matroids, Unimodularity and Multicommodity Transportation Problem", presented at the Joint ORSA/TIMS Meeting, Chicago, 1975.

3. FORD, L.R. and D.R. Fulkerson, "Flows in Networks", Princeton University Press, Princeton, N.J., 1962.

4. FORD, L.R. and D.R. Fulkerson, "A suggested computation for maximal Multicommodity Network Flows", Management Science, Vol. 5, No.1, pp. 97-101, 1958.

5. HU, T.C., "Integer Programming and Network Flows", Addison Wesley, Reading, Massachusetts, 1969.

6. JEWELL, W.S., "Multicommodity Network Solutions", Oper. Res. Center, University of California, Berkeley, 1966,

7. KARZANOV, A., "Determining the Max Flow in a Network by the Method of Preflows", Sovt. Math. Dokl., Vol. 15, pp 434-437, 1974.

8. KLEITMAN, D.J., "An Algorithm for Certain Multicommodity Flow Problems", Networks, 1, 1, 1971, pp. 75-90.

9. LAWLER, E.L., "Combinatorial Optimization: Networks and Metroids", Holt, Rinehart and Winston, 1977.

10. MURTHY, K.G., "Linear and Combinatorial Programming", John Wiley and Sons, Inc., New York, 1976.

11. ROTHFARB, B., N.P. Shein and I.T. Frisch, "Common Terminal Multicommodity Flow", Oper. Res. 16, 1, January 1968, pp. 202-205.

12. ROTHFARB, B., and I.T. Frisch, "On the Three Commodity Flow Problem", SIAM J. Appl. Math. 17, 1, January 1969, pp. 46-58.

```
C   ************************************************************
C
C   *********************    ALGORITHM-1      ******************
C         BRANCH & BOUND PROCEDURE FOR SOLVING A
C         MAXIMAL ARC-DISJOINT MULTICOMMODITY
C         SINGLE TERMINAL FLOW PROBLEM
C   ************************************************************
      IMPLICIT INTEGER (A-Z)
      COMMON NNODE,NARC,NCOMM,SORCE,SINK,START(300),FINISH(300),
     1CAP(300),NET(600),DIRECT(600),DEGREE(200),CLUE(200),
     2EDGE(600),IFLOW(300),FLOW(5,300),ASSIGN(300)
      DIMENSION IR(5)
      DIMENSION LSTCAN(50,300),SELARC(5),IASSIG(5,300),UBOUND(5)
      REAL DENSTY,ARCN,ODEM
C     ##################################################
C     #  NNODE          - Number of Nodes              #
C     #  NARC           - Number of Arcs               #
C     #  NCOMM          - Number of Commodities        #
C     #  SORCE          - Nodenumber of the super source#
C     #                     (this is equal to NNODE)    #
C     #  START(J)       - Starting node of the J th arc#
C     #  FINISH(J)      - Finishing node of the J th arc#
C     #  CAP(J)         - Capacity of the J th arc     #
C     #   NOTE                                         #
C     # ---------                                      #
C     #       - The source node of the i th commodity is
C     #             numbered 'i'                       #
C     #       - The arc connecting the super source and the #
C     #             source of the j th commodity is numbered#
C     #             'J' and the capacity of this arc is set #
C     #             to be a large number(infinity)     #
C     ##################################################
      IIN=21
      IOU=22
      OPEN (UNIT=IIN, FILE='NETGEN.IOU')
      OPEN (UNIT=IOU, FILE ='NET011.DAT')
      READ(IIN,*)NNODE,NARC,NCOMM,SINK
      READ(IIN,*)(START(J),J=1,NARC)
      READ(IIN,*)(FINISH(J),J=1,NARC)
      READ(IIN,*)(CAP(J),J=1,NARC)
      DO 3 I=1,NARC
      IFLOW(I)=0
      DO 3 J=1,NCOMM
3     FLOW(J,I)=0
      TOTCAN=0
      SORCE=NNODE
      CALL RTIME(TIME1)
C     @@@@
```

```
C   @                          - NET of nodes which are directly @
C   @                          - connected to the node I         @
C
C   @        EDGE(K)           - The edge corresponding to the   @
C   @                            K th element of NET
C   @        DIRECT(K)         -The direction of the edge        @
C   @                            corresponding to the K th       @
C
C   @                            element of NET                  @
C
C  ------------------------------------------------------------------
      K=1
      CLUE(1)=1
      DO 30 I = 1,NNODE
      DO 20 J = 1,NARC
      IF(START(J).EQ.I)GO TO 10
      IF(FINISH(J).EQ.I)GO TO 5
      GO TO 20
    5 NET(K)=START(J)
      DIRECT(K)=2
      EDGE(K)=J
      K=K+1
      GO TO 20
   10 NET(K)=FINISH(J)
      DIRECT(K)=1
      EDGE(K)=J
      K=K+1
   20 CONTINUE
      CLUE(I+1)=K
   30 DEGREE(I)=CLUE(I+1)-CLUE(I)
      INCUMB=0
      INDCAN=1
      DO 40 J=1,NARC
40  ASSIGN(J)=0
C     IF THE MAXIMUM POSSIBLE FLOW IS FEASIBLE (ARC-DISJOINT)
C     ALGORITHM TERMINATES AS THE FLOW IS THE MAXIMAL ARC-
C     DISJOINT FLOW
      JA=0
      CALL BOUND (UB,K)
      IF (K.EQ.0)GO TO998
C     IF INFEASIBLE ADDED TO THE LIST. 'K' IS THE ARC SELECTED
C     FOR BRANCHING.
50  DO 51 J=1,NARC
51  LSTCAN(INDCAN,I)=ASSIGN(J)
      LSTCAN(INDCAN,NARC+1)=K
      LSTCAN(INDCAN,NARC+2)=UB
      TOTCAN=TOTCAN+1
C     IF THE LIST IS EMPTY ALGORITHM TERMINATES
53  IF(INDCAN.EQ.0) GO TO 999
C     IF LIST IS NOT EMPTY THE LAST PROBLEM IN THE LIST IS TAKEN
      DO 52 J=1,NARC
52  ASSIGN(J)=LSTCAN(INDCAN,J)
C     IF THE UPPER BOUND IS LESS THAN THE INCUMBENT
C     THAT NODE IS PRUNED.
      IF(LSTCAN(INDCAN,NARC+2).LT.INCUMB) GO TO 54
      II=0
C     CANADITE PROBLEMS ARE GENERATED BY ALLOWING EACH OF THE
C     DIFFERENT COMMODITIES TO FLOW OVER THE ARC 'K'
      KI=LSTCAN(INDCAN,NARC+1)
      INDCAN=INDCAN-1
      DO 55 IP=1,NCOMM
      ASSIGN(KI)=IP
      DO 49 N=1,NCOMM
```

```
      DO 49 M=1,NARC
   49 FLOW(N,M)=0
      JA=0
      CALL BOUND(UB,K)
      IF (K.NE.0) GO TO 57
C     IF THE FLOW IS FEASIBLE INCUMBENT IS UPDATED IF THE
C     UPPER BOUND IS GREATER THAN THE INCUMBENT,OTHERWISE PRUNED
      IF (UB.GT.INCUMB)INCUMB=UB
      IF ((IP.EQ.NCOMM).AND.(II.EQ.0)) GO TO 53
      GO TO 55
C     INFEASIBLE PROBLEMS ARE ADDED TO THE LIST IN THE INCREASING
C     ORDER OF THE UPPER BOUND
   57 II=II+1
      DO 59 J=1,NARC
   59 IASSIG(II,J)=ASSIGN(J)
      SELARC(II)=K
      UBOUND(II)=UB
   55 CONTINUE
      DO 56 I=1,II
   56 IR(I)=I
C THE IMSL ROUTINE VSRTP IS USED IN ARRANGING IN THE PROBLEMS IN
C     THE INCREASING ORDER OF THE UPPER BOUND
      CALL VSRTP (UBOUND,II,IR)
      DO 58 J=1,I
      INDCAN=INDCAN+1
      IF (INDCAN.GT.50)GO TO 617
      TOTCAN=TOTCAN+1
      DO 60 I=1,NARC
   60 LSTCAN(INDCAN,I)=IASSIG(IR(J),I)
      LSTCAN(INDCAN,NARC+1)=SELARC(IR(J))
   58 LSTCAN(INDCAN,NARC+2)=UBOUND(J)
      GO TO 53
   54 INDCAN=INDCAN-1
      GO TO 53
  998          INCUMB=UB
  999          NODEN=NNODE-1
      IARCN=NARC-NCOMM
      CALL RTIME(TIME2)
      MSEC=TIME2-TIME1
      ARCN=IARCN
      ODEN=NODEN
      DENSTY=ARCN/(ODEN*(ODEN-1))
      WRITE(IOU,794)NODETI,NODEN,IARCN,NCOMM,INCUMB,TOTCAN,MSEC,DENSTY
  794          FORMAT(/5X,7I8,5X,F6.3)
  618          CLOSE (UNIT=IIN)
      CLOSE (UNIT=IOU)
      STOP
      END
C     ************************************************************
CC    This routine finds the max.flow and check for the feasibility
CC    of the flow
C     FLOW(K,I)          - Flow of the Kth commodity over the I th arc
CC    UB                 - Maximal flow
CC    K                  - The arc selected for branching<ie.,of the arcs
CC                         which allow more than one commodity to flow,
CC                         the arc which is having the largest flow>
C     ------------------------------------------------------------
      SUBROUTINE BOUND (UB,K)
      IMPLICIT INTEGER (A-Z)
      COMMON NNODE,NARC,NCOMM,SORCE,SINK,START(300),FINISH(300),
```

```
     1CAP(300),NET(600),DIRECT(600),DEGREE(200),CLUE(200),
     2EDGE(600),TFLOW(300),FLOW(5,300),ASSIGN(300)
     DIMENSION L1(200),L2(200),
     3L3(200),L4(200),L5(200),LIST(200)
C    ------------------------------------------------------------
C    LIST              - THE VECTOR OF NODES WHICH ARE LABELLED
C    NLIST    -NUMBER OF NODES WHICH ARE LABELLED
C    COUNT    -NUMBER OF NODES WHICH ARE LABELLED AND SCANNED
C    L1(I)    -LABEL OF THE I TH NODE INDICATING THE AMOUNT
C                  OF AUGUMENTATION POSSIBLE
C    L2(I)    -LABEL FOR THE I TH NODE INDICATING THE NODE FROM
C                  WHICH IT GOT LABELLED
C    L3(I)    -LABEL FOR THE I TH NODE INDICATING THE DIRECTION
C                  OF THE ARC BY WHICH NODE I GOT LABELLED
C                  1 IF ARC IS FORWARD
C                  2 IF ARC IS BACKWARD
C    L4(I)    -LABEL OF THE I TH NODE INDICATING THE ARC BY WHICH
C                  NODE I GOT LABELLED
C    L5(I)    -LABEL OF THE I TH NODE INDICATING THE COMMODITY
C    ------------------------------------------------------------
     LARGE=9999
     NB=0
     DO 7 I=1,NARC
7    TFLOW(I)=0
     DO 10 L=1,NCOMM
     L1(L)=LARGE
     L2(L)=SORCE
     L3(L)=1
     L4(L)=L
     L5(L)=L
     LIST(L)=L
10   CONTINUE
12   NLIST=NCOMM
C    ALL THE SOURCE NODES ARE INCLUDED IN THE LIST OF
C    LABELLED NODES AND THE SUPER SOURCE IS BOTH
C    LABELLED AND SCANNED
13   DO 14 I=NCOMM+1,NNODE-1
     L1(I)=0
     L2(I)=0
     L3(I)=0
     L4(I)=0
     L5(I)=0
     LIST(I)=0
14   CONTINUE
     COUNT=0
C       THE NODE WHICH WAS FIRST LABELLED BUT STILL UNSCANNED IS CHOUSE
C          FOR FURTHER LABELLING
15   KK=LIST(COUNT+1)
     IF(KK.EQ.0) GO TO 100
C    IF NO FURTHER LABELLING IS POSSIBLE MAX FLOW IS REACHED
18   JJ=CLUE(KK)
     LL=CLUE(KK)+DEGREE(KK)-1
     DO 50 MM=JJ,LL
     II=NET(MM)
     IF (L1(II).GT.0) GO TO 50
     NN=EDGE(MM)
     IF((ASSIGN(NN).NE.0).AND.(ASSIGN(NN).NE.L5(KK))) GO TO 50
     IF ((DIRECT(MM).EQ.1).AND.(CAP(NN).GT.TFLOW(NN)))GO TO 30
     IF ((DIRECT(MM).EQ.2).AND.(TFLOW(NN).GT.0))GO TO 40
     GO TO 50
```

```
C   IF FURTHER LABELLING IS NOT POSSIBLE JUST BECAUSE
C   THE ASSIGNMENT OF THE ARC IS DIFFERENT,THAT ARC IS
C   STORED SO THAT LATER THE COMMODITY TO WHICH THAT ARC
C   IS ASSIGNED TO COULD BE TRIED WITH
C   LABELLING USING FORWARD ARCS IF ADDITIONAL CAPACITTE IS
C             AVAILABLE
30  DIFF=CAP(NN)-TFLOW(NN)
    L1(II)=L1(KK)
    IF(DIFF.LT.L1(KK)) L1(II)=DIFF
    L2(II)=KK
    L3(II)=1
    L4(II)=NN
    L5(II)=L5(KK)
    GO TO 45
C   LABELLIG USING BACKWARD ARCS IF FLOW OVER THE ARC IS >0
40  L1(II)=L1(KK)
    IF (TFLOW(NN).LT.L1(KK)) L1(II)=TFLOW(NN)
    L2(II)=KK
    L3(II)=2
    L4(II)=NN
    L5(II)=L5(KK)
45  IF(L1(SINK).GT.0)GO TO 55
C   IF SINK IS LABELLED FLOW IS AUGUMENTED BY AMOUNT L1(SINK)
    DO 47 I=1,NLIST
    IF(II.EQ.LIST(I)) GO TO 50
47  CONTINUE
    NLIST=NLIST+1
    LIST(NLIST)=II
50  CONTINUE
    COUNT=COUNT+1
    GO TO 15
55  CHANGE=L1(SINK)
    UB=UB+CHANGE
    L=SINK
60  IF(L.EQ.SORCE)GO TO 12
    M=L4(L)
    N=L5(L)
    IF(L3(L).EQ.2)GO TO65
C   FOR FORWARD ARCS THE FLOW IS INCREASED AND FOR BACKWARD
C   ARCS THE FLOW IS DECREASED
    FLOW(N,M)=FLOW(N,M)+CHANGE
    TFLOW(M)=TFLOW(M)+CHANGE
    L=L2(L)
    GO TO 60
65          FLOW(N,M)=FLOW(N,M)-CHANGE
    TFLOW(M)=TFLOW(M)-CHANGE
    L=L2(L)
    GO TO 60
100         IF (JA.EQ.NCOMM) GO TO 70
    JA=JA+1
    NLIST=1
    LIST(1)=JA
    DO 11 IJ=2,NCOMM
11  LIST(IJ)=0
    GO TO 13
C   THIS PORTION OF THE PROGRAM IS FOR SELECTING THE ARC FOR
C   BRANCHING
70  MEDIAT=0
    K=0
    DO 80 M=NCOMM+1,NARC
```

```
      IF(TFLOW(M).EQ.0) GO TO 80
      DO 90 N=1,NCOMM
      IF (FLOW(N,M).EQ.TFLOW(M)) GO TO 80
      IF(MEDIAT.GT.TFLOW(M))GO TO 80
90    CONTINUE
      MEDIAT=TFLOW(M)
      K=M
80    CONTINUE
            RETURN
      END
```

```
C  ************************************************************
C
C           &&&&&&&&    ALGORITHM-2    &&&&&&&&
C
C           BRANCH & CLUB PROCEDURE FOR SOLVING A
C           CAPACITAL ARC-DISJOINT AUGMENTED MULTICOMMODITY
C           SINGLE TERMINAL FLOW PROBLEM
C  ************************************************************
      IMPLICIT INTEGER (A-Z)
      COMMON /CODE/,NARC,NCOMM,SORCE,SINK,START,FINISH,CAP
     1,NET,DIRECT,DEGREE,CLUB,EDGE, IFLOW ,FLOW
     1,BRANCH START(300),FINISH(300),CAP(300),NET(600),DIRECT(600),
     1DEGREE(200),CLUB(200),EDGE(600),LIST(300,2),NODE,NARC,
     2SORCE,NCOMM,SINK,FLOW(300),FLOW(5,300),IR(5),ITFLOW(300)
     3,LSTCAP(50,5,300),SAVARC(5),CASSIG(5,6,300),NCOMD(5)
     4,IFLOW(5,300),ASSIG (300)
      REAL DENSITY,NODE,,ARC,
C  ##########################################################
C  #   NNODE          - Number of Nodes                     #
C  #   NARC           - Number of Arcs                      #
C  #   NCOMM          - Number of Commodities               #
C  #   SORCE          - Nodenumber of the super source      #
C  #                      (this is equal to NNODE)          #
C  #   START(J)       - Starting node of the J th arc       #
C  #   FINISH(J)      - Finishing node of the J th arc      #
C  #   CAP(J)         - Capacity of the J th arc            #
C  #    NOTE                                               #
C  # ---------                                             #
C  #        - The source node of the i th commodity is     #
C  #             numbered 'i'                              #
C  #        - The arc connecting the super source and the  #
C  #             source of the i th commodity is numbered#
C  #             'J' and the capacity of this arc is set  #
C  #             to be a large number(infinity)           #
C  ##########################################################
      IIN=21
      IOU=22
      OPEN (UNIT=IIN, FILE='NETG8.IOU')
      OPEN (UNIT=IOU, FILE ='NET012.DAT')
      TOTCOD=0
      READ(IIN,*)NNODE,NARC,NCOMM,SINK
      READ(IIN,*)(START(J),J=1,NARC)
      READ(IIN,*)(FINISH(J),J=1,NARC)
      READ(IIN,*)(CAP(J),J=1,NARC)
      DO 3 I=1,NARC
      IFLOW(I)=0
      DO 3 J=1,NCOMM
3     FLOW(J,I)=0
      LARGE=9999
      CALL RTIME(TIME1)
C  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
C  @       This part of the program ,using the input data  @
C  @       generates the following variables which provide @
C  @       a better data struture to the program           @
C  @       ________________________________________________ @
C  @       NET(K)  - K th element of the vector of         @
C  @                    nodes NET                          @
C  @       DEGREE(I)       - The total degree of each node @
                                                  21
```

```
C   @                                   <ie.total no. of incoming    @
C   @                                        arcs+outgoing arcs>     @
C   @        CLUE(I)            = The starting point in the vctr@
C   @                             = SET of nodes which are directly
C   @                             = connected to the node I       @
C   @        EDGE(K)            = The edge corresponding to the @
C   @                             K th element of SET
C   @        DIRECT(K)          =The direction of the edge      @
C   @                             corresponding to the K th      @
C   @                             element of SET                 @
C   ------------------------------------------------------------------
      K=1
      CLUE(1)=1
      SOURCE=NNODE
      DO 30 I = 1,NNODE
      DO 20 J = 1,NARC
      IF(START(J).EQ.I)GO TO 10
      IF(FINISH(J).EQ.I)GO TO 5
      GO TO 20
    5       NET(K)=START(J)
      DIRECT(K)=2
      EDGE(K)=J
      K=K+1
      GO TO 20
   10       NET(K)=FINISH(J)
      DIRECT(K)=1
      EDGE(K)=J
      K=K+1
   20       CONTINUE
      CLUE(I+1)=K
   30       DEGREE(I)=CLUE(I+1)-CLUE(I)
      INCUM=0
      INDCAN=1
      UB=0
C     IF THE MAXIMUM POSSIBLE FLOW IS FEASIBLE (ARC-DISJOINT)
CC    ALGORITHM TERMINATES AS THE FLOW IS THE MAXIMAL ARC-
C     DISJOINT FLOW
      ISET=0
      DO 34 J=1,NARC
   34 ASSIGN(J)=0
      JA=0
      CALL BOUND(UB,IK,ISET,ASSIGN)
      IF (IK.EQ.0) GO TO 995
      TOTNOD=TOTNOD+1
C     IF INFEASIBLE ADDED TO THE LIST. 'K1' IS THE ARC SELECTED
C     FOR BRANCHING.
      DO 35 J=1,NARC
      LSTCAN(INDCAN,6,J)=ASSIGN(J)
      DO 35 K=1,NCOMM
   35 LSTCAN(INDCAN,K,J)=FLOW(K,J)
      LIST(INDCAN,1)=IK
      LIST(INDCAN,2)=UB
C     IF THE LIST IS EMPTY ALGORITHM TERMINATES
   40 IF(INDCAN.EQ.0)GO TO 999
C     IF LIST IS NOT EMPTY THE LAST PROBLEM IN THE LIST IS TAKEN
      DO 45 J=1,NARC
      ASSIGN(J)=LSTCAN(INDCAN,6,J)
      TFLOW(J)=0
      DO 45 K=1,NCOMM
      FLOW(K,J)=LSTCAN(INDCAN,K,J)
```

```
45 TFLOW(J)=TFLOW(J)+FLOW(K,J)
   IF (LIST(INDCAN,2).GE.INCUM) GO TO 50
   KI=LIST(INDCAN,1)
   UB=LIST(INDCAN,2)-TFLOW(KI)
   INDCAN=INDCAN-1
C  ALL THE FLOW PATHS WHICH CONTAIN THE ARC 'KI' ARE REMOVED
C  FROM THE FLOW BY THE ROUTINE 'PUSHBO'
   CALL PUSHBO(KI)
   INDEX=0
   DO 65 I=1,NARC
   ITFLOW(I)=TFLOW(I)
   DO 65 IKI=1,NCOMM
65 IFLOW(IKI,I)=FLOW(IKI,I)
   IUB=UB
C  CANADITE PROBLEMS ARE GENERATED BY ADDING EACH OF THE
C  DIFFERENT COMMODITIES TO FLOW OVER THE ARC 'KI'
   DO 55 IP=1,NCOMM
   ASSIG(KT)=IP
   UB=IUB
   DO 72 I=1,NARC
   TFLOW(I)=ITFLOW(I)
   DO 72 IKI=1,NCOMM
72 FLOW(IKI,I)=IFLOW(IKI,I)
   CAPKI=CAP(KI)
   ISET=1
   K=IP
   JA=0
   CALL BOUND(UB,K,ISET,ASSIGN)
   IK=IP
   CAP(KI)=0
   ISET=2
   JA=0
   CALL BOUND(UB,IK,ISET,ASSIGN)
   CAP(KI)=CAPKI
C  IF THE FLOW IS FEASIBLE INCUMBENT IS UPDATED IF THE
C  UPPER BOUND IS GREATER THAN THE INCUMBENT,OTHERWISE PRUNED
   IF (IK.NE.0) GO TO 75
   IF (UB.GT.INCUM) INCUM=UB
   IF((IP.EQ.NCOMM).AND.(INDEX.EQ.0)) GO TO 40
   GO TO 55
C  INFEASIBLE PROBLEMS ARE ADDED TO THE LIST IN THE INCREASING
C  ORDER OF THE UPPER BOUND
75 INDEX=INDEX+1
   DO 80 J=1,NARC
   IASSIG(INDEX,0,J)=ASSIGN(J)
   DO 80 K=1,NCOMM
80 IASSIG(INDEX,K,J)=FLOW(K,J)
   SELARC(INDEX)=IK
   UBOUND(INDEX)=UB
55 CONTINUE
   DO 85 I=1,INDEX
85 IR(I)=I
C THE IMSL ROUTINE VSRTPM IS USED IN ARRANGING IN THE PROBLEMS IN
C  THE INCREASING ORDER OF THE UPPER BOUND
   CALL VSRTPM(UBOUND,INDEX,IR)
   DO 90 J=1,INDEX
   INDCAN=INDCAN+1
   IF(INDCAN.GT.50) GO TO 617
   TOTNOD=TOTNOD+1
   DO 95 I=1,NARC
```

```
      LSTCAN(INDCAN,6,I)=IASSIG(IN(J),6,I)
      DO 95 K=1,NCOMM
95    LSTCAN(INDCAN,K,I)=IASSIG(IN(J),K,I)
      LIST(INDCAN,1)=SEGARC(IN(J))
90    LIST(INDCAN,2)=NBOUND(J)
      GO TO 40
50    INDCAN=INDCAN-1
      GO TO 40
998           TNCOM=NC
999           NODEM=NNODE-1
      IARCM=NARC-NCOMM
      CALL RTIME(ITIME2)
      MSEC=ITIME2-ITIME1
      ARCM=IARCM
      ODEM=NODEM
      DENSTY=ARCM/(ODEM*(ODEM-1))
      WRITE(IOU,794)NCOMM,NODEM,IARCM,NCOMM,IPCM,PRTNUM,ARC,DENSTY
794           FORMAT(/5X,7I8,5X,F6.3)
618           CLOSE(UNIT=IIN)
      CLOSE(UNIT=IOU)
      STOP
      END
C     ***************************************************************
C     This routine finds the max.flow and check for the feasibility
C     of the flow
C     FLOW(K,I)          - Flow of the Kth commodity over the I th arc
C     UB                 - Maximal flow
C     K                  - The arc selected for branching<ie.,of the arcs
C                          which allow more than one commodity to flow,
C                          the arc which is having the largest flow>
C     ---------------------------------------------------------------
      SUBROUTINE BOUND (UB,K,ISET,ASSIG)
      IMPLICIT INTEGER (A-Z)
      COMMON NNODE,NARC,NCOMM,SORCE,SINK,START,FINISH,CAP
     1,NET,DIRECT,DEGREE,CLOSE,EDGE, TFLOW ,FLOW
      INTEGER SORCE,SINK,ENODE,NCOMM, ARC,START(300),FINISH(300),
     1CAP(300),NET(600),DIRECT(600),DEGREE(200),CLOSE(200),
     2EDGE(600),ASSIGN(300),FLOW(5,300),TFLOW(300),L1(200),L2(200),
     3L3(200),L4(200),L5(200), LIST(200),COUNT,IA(10),IK(10)
C     ---------------------------------------------------------------
C     LIST               - THE VECTOR OF NODES WHICH ARE LABELLED
C     NLIST    -NUMBER OF NODES WHICH ARE LABELLED
C     COUNT    -NUMBER OF NODES WHICH ARE LABELLED AND SCANNED
C     L1(I)    -LABEL OF THE I TH NODE INDICATING THE AMOUNT
C                OF AUGMENTATION POSSIBLE
C     L2(I)    -LABEL FOR THE I TH NODE INDICATING THE NODE FROM
C                WHICH IT GOT LABELLED
C     L3(I)    -LABEL FOR THE I TH NODE INDICATING THE DIRECTION
C                OF THE ARC BY WHICH NODE I GOT LABELLED
C                1 IF ARC IS FORWARD
C                2 IF ARC IS BACKWARD
C     L4(I)    -LABEL OF THE I TH NODE INDICATING THE ARC BY WHICH
C                NODE I GOT LABELLED
C     L5(I)    -LABEL OF THE I TH NODE INDICATING THE COMMODITY
C     ---------------------------------------------------------------
      LARGE=9999
12    DO 10 L=1,NCOMM
      L1(L)=LARGE
      L2(L)=SORCE
      L3(L)=1
```

```
      L4(L)=L
      L5(L)=L
   10 CONTINUE
      NLIST=NCOMM
C     ALL THE SOURCE NODES ARE INCLUDED IN THE LIST OF
C     LABELLED NODES AND THE SUPER SOURCE IS NOTED
C     LABELLED AND SCANNED
   13 DO 14 I=NCOMM+1,NNODE-1
      L1(I)=0
      L2(I)=0
      L3(I)=0
      L4(I)=0
      L5(I)=0
      LIST(I)=0
   14 CONTINUE
      COUNT=0
      IF(ISET.EQ.1) GO TO 102
      IF(ISET.EQ.2)GO TO 101
      DO 11 L=1,NCOMM
   11 LIST(L)=L
C        THE NODE WHICH WAS FIRST LABELLED BUT STILL UNSCANNED IS CHOSEN
C           FOR FURTHER LABELLING
   15 KK=LIST(COUNT+1)
      IF (KK.EQ.0) GO TO 100
C     IF NO FURTHER LABELLING IS POSSIBLE MAX FLOW IS REACHED
   18 JJ=CLUE(KK)
      LL=CLUE(KK)+DEGREE(KK)-1
      DO 50 MM=JJ,LL
      II=NET(MM)
      IF (L1(II).GT.0) GO TO 50
      NN=EDGE(MM)
      IF(CAP(NN).EQ.0) GO TO 50
      IF (ASSIGN(NN).NE.0.AND.ASSIGN(NN).NE.LS(KK)) GO TO 50
      IF ((DIRECT(MM).EQ.1).AND.(CAP(NN).GT.TFLOW(NN)))GO TO 30
      IF ((DIRECT(MM).EQ.2).AND.(TFLOW(NN).GT.0))GO TO 40
      GO TO 50
C     LABELLING USING FORWARD ARCS IF ADDITIONAL CAPACITY IS
C              AVALIBLE
   30 DIFF=CAP(NN)-IFLOW(NN)
      L1(II)=L1(KK)
      IF(DIFF.LT.L1(KK)) L1(II)=DIFF
      L2(II)=KK
      L3(II)=1
      L4(II)=NN
      L5(II)=LS(KK)
      GO TO 45
C     LABELLIG USING BACKWARD ARCS IF FLOW OVER THE ARC IS >0
   40 L1(II)=L1(KK)
      IF (TFLOW(NN).LT.L1(KK)) L1(II)=TFLOW(NN)
      L2(II)=KK
      L3(II)=2
      L4(II)=NN
      L5(II)=L5(KK)
   45 IF(KK.EQ.SORCE)L5(II)=II
C     IF SINK IS LABELLED FLOW IS AUGUMENTED BY AMOUNT L1(SINK)
      IF(L1(SINK).GT.0)GO TO 55
      DO 47 I=1,NLIST
      IF(II.EQ.LIST(I)) GO TO 50
   47 CONTINUE
      NLIST=NLIST+1
```

```
      LIST(NLIST)=II
50    CONTINUE
57    COUNT=COUNT+1
      GO TO 15
55    CHANGE=L1(SINK)
      UB=UB+CHANGE
      L=SINK
60    IF(L.EQ.SORCE) GO TO 12
      M=L4(L)
      N=L5(L)
C     FOR FORWARD ARCS THE FLOW IS INCREASED AND FOR BACKWARD
C     ARCS THE FLOW IS DECREASED
      IF(L3(L).EQ.2)GO TO 65
      FLOW(M,N)=FLOW(M,N)+CHANGE
      TFLOW(M)=TFLOW(M)+CHANGE
      L=L2(L)
      GO TO 60
65              FLOW(M,N)=FLOW(M,N)-CHANGE
      TFLOW(M)=TFLOW(M)-CHANGE
      L=L2(L)
      GO TO 60
101             NLIST=NCOMM-1
      LL=0
      DO 103 L=1,NCOMM
      IF (L.EQ.K) GO TO 103
      LL=LL+1
      LIST(LL)=L
103             CONTINUE
      GO TO 15
102             NLIST=1
      LIST(1)=K
      GO TO 15
C     THIS PORTION OF THE PROGRAM IS FOR SELECTING THE ARC FOR
C     BRANCHING
100             IF(JA.EQ.NCOMM)GO TO 70
      JA=JA+1
      NLIST=1
      LIST(1)=JA
      DO 111 II=2,NCOMM
111             LIST(I)=0
      GO TO 13
70    MEDIAN=0
      K=0
      DO 80 M=NCOMM+1,NARC
      IF(IFLOW(M).EQ.0) GO TO 80
      DO 90 N=1,NCOMM
      IF (FLOW(M,N).EQ.IFLOW(M)) GO TO 80
      IF(MEDIAN.GT.IFLOW(M))GO TO 90
90    CONTINUE
      MEDIAN=IFLOW(M)
      K=M
80    CONTINUE
              RETURN
      END
C     %%%%
```

```
     1,NET,DIRECT,DEGREE,CLUE,EDGE,  IFLOW,  FLOW
     INTEGER START(300),FINISH(300),COST(300),CAP(600),DIRECT(600),
     1DEGREE(200),CLUE(200),EDGE(600),SOURCE,SINK,
     2SORCE,NCOMM,STNK,IFLOW(300),FLOW(5,300),ISAVE(6:300) ,CHANGE
     DO 5 KCOM=1,NCOMM
     IF(FLOW(KCOM,K).EQ.0) GO TO 5
     CHANGE=0
     LIM=FLOW(KCOM,K)
     IREDU(1)=K
     IFLOW(KCOM)=IFLOW(KCOM)-FLOW(KCOM,K)
     FLOW(KCOM,KCOM)=FLOW(KCOM,KCOM)-FLOW(KCOM,K)
     IF(START(K).NE.KCOM)GO TO 10
     CHANGE=CHANGE+FLOW(KCOM,K)
     GO TO 40
10   IAMT=LIM-CHANGE
     II=1
     NN=K
15   KK=START(NN)
     JJ=CLUE(KK)
     LL=CLUE(KK)+DEGREE(KK)-1
     DO 20 MM=JJ,LL
     NN=EDGE(MM)
     IF(DIRECT(MM).EQ.1)GO TO 20
     DO 81 IIK=1,II
81   IF(NN.EQ.IREDU(IIK)) GO TO 20
     IF((FLOW(KCOM,NN).NE.0).AND.(FINISH(NN).EQ.START(IREDU(II))))
     1  GO TO 30
20   CONTINUE
30   IF(FLOW(KCOM,NN).LT.IAMT)IAMT=FLOW(KCOM,NN)
     IF(IAMT.LE.0) GO TO 40
     II=II+1
     IREDU(II)=NN
     IF(START(NN).NE.KCOM) GO TO 15
42   CHANGE=CHANGE+IAMT
     DO 50 I=2,II
     IFLOW(IREDU(I))=IFLOW(IREDU(I))-IAMT
50   FLOW(KCOM,IREDU(I))=FLOW(KCOM,IREDU(I))-IAMT
     IF(CHANGE.LT.LIM) GO TO 10
40   FLOW(KCOM,K)=LIM
     CHANGE=0

     IF(FINISH(K).NE.SINK) GO TO 41
     CHANGE=LIM
     GO TO 6
41   IAMT=LIM-CHANGE
     II=1
     NN=K
45   KK=FINISH(NN)
     JJ=CLUE(KK)
     LL=CLUE(KK)+DEGREE(KK)-1
     DO 35 MM=JJ,LL
     NN=EDGE(MM)
     IF(DIRECT(MM).EQ.2)GO TO  35
     DO 82 IIK=1,II
82   IF(NN.EQ.IREDU(IIK))GO TO 35
     IF((FLOW(KCOM,NN).NE.0).AND.(START(NN).EQ.FINISH(IREDU(II))))
     1  GO TO 51
35   CONTINUE
51   IF(FLOW(KCOM,NN).LT.IAMT)IAMT=FLOW(KCOM,NN)
     IF(IAMT.LE.0) GO TO 6
```

```
      TI=IT+1
      IREDU(TI)=N
      IF(FINISH(N).NE.STOK) GO TO 45
60    CHANGE=CHANGE+IAMT
      DO 70 I=1,TI
      TFLOW(IREDU(I))=TFLOW(IREDU(I))-IAMT
70    FLOW(KCOM,IREDU(I))=FLOW(KCOM,IREDU(I))-IAMT
      IF(CHANGE.EQ.0) GO TO 4
8     FLOW(KCOM,K)=0
5     CONTINUE
      TFLOW(K)=0
      RETURN
      END
```

```
00100 C     ************************************************************
00200 C
00300 C           PROGRAMME FOR THE GENERATION
00400 C           OF NETWORKS WHICH ARE CONNECTED
00500 C
00600 C     ************************************************************
00700       DIMENSION INCID(200,200),R(300)
00800       INTEGER STAPT(350),FINISP(350),CAP(350),TITL
00900       IOU=22
01000       ILV=21
01100       OPEN (UNIT=ILV,FILE='.FIGEN.INP')
01200       OPEN (UNIT=IOU,FILE='.FIGEN.OUP')
01300
01400 C     IC  Provides the seed for random number generation
01500 C           done in the subroutine RANGEN
01600 C
01700 C     N   Is the number of random numbers generated at
01800 C           one time.If the requirement of random
01900 C           numbers is more than N , the subroutine
02000 C           RANGEN is again executed.
02100 C     R     Is an array of N random numbers generated.
02200
02300
02400       IC=36794
02500       N=300
02600       CALL RANGEN(IC,N,R,INDEX)
02700
02800 C     NNODE  is the number of nodes required in the netw
02900 C     NARC   is the number of arcs required in the netw
03000 C     NCOMM  is the number of commodities in the proble
03100 C     MAXCAP is the maximum capacity allowed for any arc
03200 C
03300       DO 3001 NUMTIM=1,5
03400       READ(IIN,*) NNODE,NARC,,MAXCAP
03500       DO 3001 NCOMM=2,5
03600
03700 C     INCID  is the node-node incidence matrix
03800
03900       DO 5 I=1,NNODE
04000       DO 5 J=1,NNODE
04100     5 INCID(I,J)=0
04200 C     ZMUE gives the average total degree of arcs
04300 C     IOTDEG generates randomly the degree of any node
04400
04500       ZMUE=2*NARC/NNODE
04600       DO 35 I=1,NNODE
04700       IOTDEG=R(INDEX)*2*ZMUE
04800       INDEX=INDEX+1
04900       IF (INDEX.EQ.N)CALL RANGEN(IC,N,R,INDEX)
05000       IF(IOTDEG.LT.2)IOTDEG=2
05100
05200 C     INDEG  is the randomly generated indegree of
05300 C              any node
05400 C     IOUDEG  is the randomly generated outdegree
05500 C              of any node
05600
05700       INDEG=IOTDEG*R(INDEX)
05800       INDEX=INDEX+1
```

```
05900                 IF (INDEX.EQ.N)CALL RANGEN(IC,N,R,INDEX)
06000                 IF(INDEG.LT.1)INDEG=1
06100                 IF (INDEG.EQ.IOTDEG)INDEG=INDEG-1
06200                 IOUDEG=IOTDEG-INDEG
06300                 IF(I.LE.NCOMM)GO TO 25
06400                 TP=0
06500                 DO 15 J=1,NNODE
06600                 IF(INCID(J,1).GT.0) TP=TP+1
06700           15    CONTINUE
06800                 DO 25 M=1,INDEG-TP
06900                 ITME=0
07000
07100           C     ICAP  is the randomly generated capacities for
07200           C           different arcs stored in randomly generated
07300           C           locations of the node-node incidence matrix
07400
07500           30    K=R(INDEX)*NNODE
07600                 INDEX=INDEX+1
07700                 IF (INDEX.EQ.N)CALL RANGEN(IC,N,R,INDEX)
07800                 IF((K.LE.NCOMM).OR.(I.EQ.K))GO TO 30
07900           32    ICAP=R(INDEX)*MAXCAP
08000                 ITME=ITME+1
08100                 INDEX=INDEX+1
08200                 IF (INDEX.EQ.N)CALL RANGEN(IC,N,R,INDEX)
08300                 IF(ICAP.EQ.0)GO TO 32
08400                 IF((INCID(K,I).NE.0).AND.(ITME.LE.NNODE))GO TO 30
08500                 INCID(K,I)=ICAP
08600                 INCID(I,K)=-ICAP
08700           25    CONTINUE
08800                 IF(I.EQ.NNODE)GO TO 35
08900                 TP=0
09000                 DO 20 J=1,NNODE
09100                 IF(INCID(I,J).GT.0)TP=TP+1
09200           20    CONTINUE
09300                 DO 35 M=1,IOUDEG-TP
09400                 ITME=0
09500           40    K=R(INDEX)*NNODE
09600                 INDEX=INDEX+1
09700                 IF (INDEX.EQ.N)CALL RANGEN(IC,N,R,INDEX)
09800                 IF ((K.LE.NCOMM).OR.(I.EQ.K)) GO TO 40
09900                 ITME=ITME+1
10000           42    ICAP=R(INDEX)*MAXCAP
10100                 INDEX=INDEX+1
10200                 IF (INDEX.EQ.N)CALL RANGEN(IC,N,R,INDEX)
10300                 IF(ICAP.EQ.0) GO TO 42
10400                 IF((INCID(K,I).NE.0).AND.(ITME.LE.NNODE))GO TO 40
10500                 INCID(I,K)=ICAP
10600                 INCID(K,I)=-ICAP
10700           35    CONTINUE
10800
10900           C     From the node-node incidence matrix ,the input data
11000           C     is  generated
11100
11200           C     KNODE  is the total number of nodes including the
11300           C            super source
11400           C     NUMARC   is the total number of arcs including the
11500           C            arcs connecting the super source and sources
11600           C     START(J)  is the starting node of the Jth arc
11700           C     FINISH(J)  is the finishing point of the Jth arc
11800           C     CAP(J)  is the capacity of the Jth arc
```

```
11900
12000                 NUMARC=NCOMM
12100                 DO 110 I=1,NNODE
12200                 DO 110 J=1,NNODE
12300                 IF (IACTD(I,J).EQ.0)GO TO 110
12400                 NUMARC=NUMARC+1
12500                 START(NUMARC)=I
12600                 FINISH(NUMARC)=J
12700                 CAP(NUMARC)=IACTD(I,J)
12800           110   CONTINUE
12900                 DO 120 I=1,NCOMM
13000                 START(I)=NNODE+1
13100                 FINISH(I)=I
13200                 CAP(I)=9999
13300           120   CONTINUE
13400                 KNODE=NNODE+1
13500                 IF(NUMARC.GT.300)GO TO 3001
13600                 WRITE(IOU,*) KNODE,NUMARC,NCOMM,SLOOP
13700                 WRITE(IOU,*)(START(J),J=1,NUMARC)
13800                 WRITE(IOU,*)(FINISH(J),J=1,NUMARC)
13900                 WRITE(IOU,*)(CAP(J),J=1,NUMARC)
14000           3001  CONTINUE
14300                 CLOSE (UNIT=IIN)
14400                 CLOSE (UNIT=IOU)
14500                 STOP
14600                 END
14700
14800           C     <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
14900           C           This subroutine is used for the
15000           C           generation of random numbers
15100           C     >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
15200
15300           C     This routine uses the IMSL subroutine GGUB
15400           C           which generates random numbers.
15500
15600                 SUBROUTINE RANGEN(IC,N,R,INDEX)
15700                 DIMENSION R(300)
15800                 IC=IC+387
15900                 CALL GGUB(IC,N,R)
16000                 INDEX=1
16100                 RETURN
16200                 END
```